Daniel Nedrow

CEG 3110 Project 1

**Some Notes and Assumptions for Test Plan**

Test Case 1 is designed to test the randomness of the outcomes for tossing a single die many times. We want each number to come up 1/6 of the time, or about 16.667%. Since randomness does not produce such exact outcomes, I’ve decided that for a massive sample each outcome should be within 0.008% of the desired result.

Test Case 2 is designed to test for conditional frequency. We will again roll a single die some massive number of times, and then have our software go and examine the outcomes that come after each possible initial result of a die roll between 1 and 6. For example, given a roll of 3, what percentage do each of the 6 numbers come up on the next roll? We will allow for a somewhat larger margin here because the sample size is being divided into 6 cases. I will allow for each outcome being within 0.048% (6 times the allowance in Test Case 1) of the desired result.

Test Case 3 simply rolls 2 dice some massive number of times and counts how often doubles came up. The expected result should be 1/6 of the rolls. We will roll half as many trials as Test Case 1 and allow a margin that is twice as large. I will accept an answer that is within 0.016% of the desired result.

Test Case 4 also rolls 2 dice a massive number of times, but now we are checking the frequency that every possible sum of 2 dice comes up (sums can range from 2 to 12). The sums have a normal distribution as follows: 2 and 12 come up 1/36 times, 3 and 11 come up 2/36 times, 4 and 10, come up 3/36 times, 5 and 9 come up 4/36 times, 6 and 8 come up 5/36 times, and 7 comes up 6/36 times. As in the previous test case, we will allow results that are within 0.016% of each desired result.

Test Case 5 is essentially a double trial of Test Case 1 where we want to make sure that the distributions of outcomes from one trial to the next are not identical. If we got identical distributions, that would show the random number generator was at best just shuffling the exact same quantity of 1’s 2’s 3’s 4’s 5’s and 6’s in every trial. We will be satisfied with any difference in outcomes here, so any non-zero number will be accepted.

**Test Plan**

**Test Case 1:**

Input (number of times to roll one die): 100000000 (100,000,000 for readability)

Expected Result: The number 1 came up [16.659, 16.675] % of the time

The number 2 came up [16.659, 16.675] % of the time

The number 3 came up [16.659, 16.675] % of the time

The number 4 came up [16.659, 16.675] % of the time

The number 5 came up [16.659, 16.675] % of the time

The number 6 came up [16.659, 16.675] % of the time

Actual Result: The number 1 came up 16.668% of the time

The number 2 came up 16.666% of the time

The number 3 came up 16.667% of the time

The number 4 came up 16.670% of the time

The number 5 came up 16.659% of the time

The number 6 came up 16.669% of the time

Pass: Fail:

**Test Case 2:**

Input (number of times to roll one die): 100000000 (100,000,000 for readability)

Expected Result: Given a roll of 1:

The number 1 came up next [16.619, 16.715] % of the time

The number 2 came up next [16.619, 16.715] % of the time

The number 3 came up next [16.619, 16.715] % of the time

The number 4 came up next [16.619, 16.715] % of the time

The number 5 came up next [16.619, 16.715] % of the time

The number 6 came up next [16.619, 16.715] % of the time

Given a roll of 2:

The number 1 came up next [16.619, 16.715] % of the time

The number 2 came up next [16.619, 16.715] % of the time

The number 3 came up next [16.619, 16.715] % of the time

The number 4 came up next [16.619, 16.715] % of the time

The number 5 came up next [16.619, 16.715] % of the time

The number 6 came up next [16.619, 16.715] % of the time

Given a roll of 3:

The number 1 came up next [16.619, 16.715] % of the time

The number 2 came up next [16.619, 16.715] % of the time

The number 3 came up next [16.619, 16.715] % of the time

The number 4 came up next [16.619, 16.715] % of the time

The number 5 came up next [16.619, 16.715] % of the time

The number 6 came up next [16.619, 16.715] % of the time

Given a roll of 4:

The number 1 came up next [16.619, 16.715] % of the time

The number 2 came up next [16.619, 16.715] % of the time

The number 3 came up next [16.619, 16.715] % of the time

The number 4 came up next [16.619, 16.715] % of the time

The number 5 came up next [16.619, 16.715] % of the time

The number 6 came up next [16.619, 16.715] % of the time

Given a roll of 5:

The number 1 came up next [16.619, 16.715] % of the time

The number 2 came up next [16.619, 16.715] % of the time

The number 3 came up next [16.619, 16.715] % of the time

The number 4 came up next [16.619, 16.715] % of the time

The number 5 came up next [16.619, 16.715] % of the time

The number 6 came up next [16.619, 16.715] % of the time

Given a roll of 6:

The number 1 came up next [16.619, 16.715] % of the time

The number 2 came up next [16.619, 16.715] % of the time

The number 3 came up next [16.619, 16.715] % of the time

The number 4 came up next [16.619, 16.715] % of the time

The number 5 came up next [16.619, 16.715] % of the time

The number 6 came up next [16.619, 16.715] % of the time

Actual Result: Given a roll of 1:

The number 1 came up next 16.667% of the time

The number 2 came up next 16.676% of the time

The number 3 came up next 16.677% of the time

The number 4 came up next 16.658% of the time

The number 5 came up next 16.657% of the time

The number 6 came up next 16.665% of the time

Given a roll of 2:

The number 1 came up next 16.681% of the time

The number 2 came up next 16.654% of the time

The number 3 came up next 16.669% of the time

The number 4 came up next 16.663% of the time

The number 5 came up next 16.656% of the time

The number 6 came up next 16.676% of the time

Given a roll of 3:

The number 1 came up next 16.675% of the time

The number 2 came up next 16.665% of the time

The number 3 came up next 16.658% of the time

The number 4 came up next 16.660% of the time

The number 5 came up next 16.671% of the time

The number 6 came up next 16.670% of the time

Given a roll of 4:

The number 1 came up next 16.654% of the time

The number 2 came up next 16.681% of the time

The number 3 came up next 16.664% of the time

The number 4 came up next 16.676% of the time

The number 5 came up next 16.667% of the time

The number 6 came up next 16.658% of the time

Given a roll of 5:

The number 1 came up next 16.666% of the time

The number 2 came up next 16.688% of the time

The number 3 came up next 16.660% of the time

The number 4 came up next 16.665% of the time

The number 5 came up next 16.663% of the time

The number 6 came up next 16.659% of the time

Given a roll of 6:

The number 1 came up next 16.673% of the time

The number 2 came up next 16.661% of the time

The number 3 came up next 16.664% of the time

The number 4 came up next 16.652% of the time

The number 5 came up next 16.678% of the time

The number 6 came up next 16.671% of the time

Pass: Fail:

**Test Case 3:**

Input (number of times to roll two dice): 50000000 (50,000,000 for readability)

Expected Result: The percent of times doubles were rolled was [16.651, 16.683] %

Actual Result: The percent of times doubles were rolled was 16.661%

Pass: Fail:

**Test Case 4:**

Input (number of times to roll two dice): 50000000 (50,000,000 for readability)

Expected Result: The sum 2 came up [2.762, 2.794] % of the time

The sum 3 came up [5.540, 5.572] % of the time

The sum 4 came up [8.317, 8.349] % of the time

The sum 5 came up [11.095, 11.127] % of the time

The sum 6 came up [13.873, 13.905] % of the time

The sum 7 came up [16.651, 16.683] % of the time

The sum 8 came up [13.873, 13.905] % of the time

The sum 9 came up [11.095, 11.127] % of the time

The sum 10 came up [8.317, 8.349] % of the time

The sum 11 came up [5.540, 5.572] % of the time

The sum 12 came up [2.762, 2.794] % of the time

Actual Result: The sum 2 came up 2.783% of the time

The sum 3 came up 5.558% of the time

The sum 4 came up 8.335% of the time

The sum 5 came up 11.110% of the time

The sum 6 came up 13.883% of the time

The sum 7 came up 16.669% of the time

The sum 8 came up 13.891% of the time

The sum 9 came up 11.109% of the time

The sum 10 came up 8.330% of the time

The sum 11 came up 5.555% of the time

The sum 12 came up 2.778% of the time

Pass: Fail:

**Test Case 5:**

Input (number of times to roll one die in each of two trials): 50000000 (50,000,000 readable)

Expected Result:

For a roll of 1, the difference in the frequency between trial 1 and trial 2 is <non-zero>

For a roll of 2, the difference in the frequency between trial 1 and trial 2 is <non-zero>

For a roll of 3, the difference in the frequency between trial 1 and trial 2 is <non-zero>

For a roll of 4, the difference in the frequency between trial 1 and trial 2 is <non-zero>

For a roll of 5, the difference in the frequency between trial 1 and trial 2 is <non-zero>

For a roll of 6, the difference in the frequency between trial 1 and trial 2 is <non-zero>

Actual Result:

For a roll of 1, the difference in the frequency between trial 1 and trial 2 is 0.007%

For a roll of 2, the difference in the frequency between trial 1 and trial 2 is 0.001%

For a roll of 3, the difference in the frequency between trial 1 and trial 2 is 0.002%

For a roll of 4, the difference in the frequency between trial 1 and trial 2 is 0.003%

For a roll of 5, the difference in the frequency between trial 1 and trial 2 is 0.001%

For a roll of 6, the difference in the frequency between trial 1 and trial 2 is 0.012%

Pass: Fail:

**Entire Sample Output of Program Run**

run:

SINGLE DIE (5 RUNS): TWO DICE (5 RUNS):

1 5 3

6 6 5

5 2 5

6 1 2

2 1 4

Test Case 1

This is a distribution test for a single die.

How many times would you like to roll the die: 100000000

The number 1 came up 16.668% of the time

The number 2 came up 16.666% of the time

The number 3 came up 16.667% of the time

The number 4 came up 16.670% of the time

The number 5 came up 16.659% of the time

The number 6 came up 16.669% of the time

Test Case 2

This is a test for independence from past events.

How many times would you like to roll the die: 100000000

Given a roll of 1:

The number 1 came up next 16.667% of the time

The number 2 came up next 16.676% of the time

The number 3 came up next 16.677% of the time

The number 4 came up next 16.658% of the time

The number 5 came up next 16.657% of the time

The number 6 came up next 16.665% of the time

Given a roll of 2:

The number 1 came up next 16.681% of the time

The number 2 came up next 16.654% of the time

The number 3 came up next 16.669% of the time

The number 4 came up next 16.663% of the time

The number 5 came up next 16.656% of the time

The number 6 came up next 16.676% of the time

Given a roll of 3:

The number 1 came up next 16.675% of the time

The number 2 came up next 16.665% of the time

The number 3 came up next 16.658% of the time

The number 4 came up next 16.660% of the time

The number 5 came up next 16.671% of the time

The number 6 came up next 16.670% of the time

Given a roll of 4:

The number 1 came up next 16.654% of the time

The number 2 came up next 16.681% of the time

The number 3 came up next 16.664% of the time

The number 4 came up next 16.676% of the time

The number 5 came up next 16.667% of the time

The number 6 came up next 16.658% of the time

Given a roll of 5:

The number 1 came up next 16.666% of the time

The number 2 came up next 16.688% of the time

The number 3 came up next 16.660% of the time

The number 4 came up next 16.665% of the time

The number 5 came up next 16.663% of the time

The number 6 came up next 16.659% of the time

Given a roll of 6:

The number 1 came up next 16.673% of the time

The number 2 came up next 16.661% of the time

The number 3 came up next 16.664% of the time

The number 4 came up next 16.652% of the time

The number 5 came up next 16.678% of the time

The number 6 came up next 16.671% of the time

Test Case 3

This is a test for doubles when rolling 2 dice.

How many times would you like to roll 2 dice: 50000000

The percent of times doubles were rolled was 16.661%

Test Case 4

This is a distribution test for the sum of 2 dice.

How many times would you like to roll 2 dice: 50000000

The sum 2 came up 2.783% of the time

The sum 3 came up 5.558% of the time

The sum 4 came up 8.335% of the time

The sum 5 came up 11.110% of the time

The sum 6 came up 13.883% of the time

The sum 7 came up 16.669% of the time

The sum 8 came up 13.891% of the time

The sum 9 came up 11.109% of the time

The sum 10 came up 8.330% of the time

The sum 11 came up 5.555% of the time

The sum 12 came up 2.778% of the time

Test Case 5

This is a test to ensure two separate distributions are not identical.

How many times would you like to roll the die each time: 50000000

For a roll of 1, the difference in the frequency between trial 1 and trial 2 is 0.007%

For a roll of 2, the difference in the frequency between trial 1 and trial 2 is 0.001%

For a roll of 3, the difference in the frequency between trial 1 and trial 2 is 0.002%

For a roll of 4, the difference in the frequency between trial 1 and trial 2 is 0.003%

For a roll of 5, the difference in the frequency between trial 1 and trial 2 is 0.001%

For a roll of 6, the difference in the frequency between trial 1 and trial 2 is 0.012%

BUILD SUCCESSFUL (total time: 1 minute 24 seconds)

**Source Code (client class)**

/\*

\* Daniel Nedrow

\* CEG 3110

\* Project 1

\* Professor John Reisner

\*/

package ceg3110hw1;

/\*\*

\* @author Daniel This class simulates a random dice roller.

\*/

public class DiceRoller {

/\*\*

\* Takes in an integer representing the number of dice to be rolled and

\* returns an array containing that number of randomly rolled dice.

\*

\* @param numDice the number of dice to be rolled (must be between 1 and 5)

\* @return an array containing the requested number of randomly rolled dice

\*/

public int[] rollDice(int numDice) {

if (numDice < 1 || numDice > 5) {

System.out.println("Error: Roll between 1 and 5 dice at a time.");

System.exit(-1);

}

int[] dice = new int[numDice];

for (int i = 0; i < numDice; i++) {

dice[i] = (int) (Math.random() \* 6 + 1);

}

return dice;

}

}

**Source Code (testing class)**

package ceg3110hw1;

import java.util.Scanner;

/\*\*

\* @author Daniel Testing class for DiceRoller.

\*/

public class CEG3110HW1 {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

Scanner keyboard = new Scanner(System.in);

DiceRoller dr = new DiceRoller();

// The following code merely demonstrates basic functionality of program

int[] rolledTwo;

System.out.printf("%S%25S\n", "Single die (5 runs):", "Two dice (5 runs):");

for (int i = 0; i < 5; i++) {

System.out.print(dr.rollDice(1)[0]);

rolledTwo = dr.rollDice(2);

System.out.printf("%27d", rolledTwo[0]);

System.out.printf("%2d\n", rolledTwo[1]);

}

testCase1(keyboard, dr); // test distribution of single die rolls

testCase2(keyboard, dr); // test that die rolls are independent from each other

testCase3(keyboard, dr); // test for percentage of doubles w/ 2 dice

testCase4(keyboard, dr); // test distribution for sum of 2 dice

testCase5(keyboard, dr); // test that 2 separate distributions are not identical

}

/\*\*

\* This method rolls a single die a number of times the user chooses. It

\* then outputs the frequency (as a percent) that each number came up.

\* @param keyboard for user input

\* @param dr the DiceRoller

\*/

public static void testCase1(Scanner keyboard, DiceRoller dr) {

System.out.println("\nTest Case 1");

System.out.println("This is a distribution test for a single die.");

System.out.print("How many times would you like to roll the die: ");

int numRolls = keyboard.nextInt();

int[] counts = new int[6];

int roll;

// roll the die the requested # of times and track counts for each outcome

for (int i = 0; i < numRolls; i++) {

roll = dr.rollDice(1)[0];

counts[roll - 1]++;

}

// calculate and output the frequency (as a percent) of each outcome

double[] percentRolled = new double[6];

for (int i = 0; i < 6; i++) {

percentRolled[i] = ((double) counts[i] / numRolls) \* 100;

System.out.printf("The number %d came up %.3f%% of the time\n",

(i + 1), percentRolled[i]);

}

}

/\*\*

\* This method rolls a single die a number of times the user chooses. It then

\* calculates and displays the conditional frequency (as a percent) of each

\* outcome, given the outcome that came before it. For example, given a roll

\* of 3, what is the percentage that 4 came up on the next roll.

\* @param keyboard for user input

\* @param dr the DiceRoller

\*/

public static void testCase2(Scanner keyboard, DiceRoller dr) {

System.out.println("\nTest Case 2");

System.out.println("This is a test for independence from past events.");

System.out.print("How many times would you like to roll the die: ");

int numRolls = keyboard.nextInt();

int[] allRolls = new int[numRolls];

int[] counts = new int[6];

// roll the die the requested number of times and track counts for each outcome

for (int i = 0; i < numRolls; i++) {

allRolls[i] = dr.rollDice(1)[0];

counts[allRolls[i] - 1]++;

}

// given any particular roll, count the outcomes of the next roll

int[][] conditionalCounts = new int[6][6];

for (int i = 0; i < numRolls - 1; i++) {

// conditionalCounts[outcomes 1-6 this roll][outcomes of next roll]

conditionalCounts[allRolls[i] - 1][allRolls[i + 1] - 1]++;

}

// calculate the conditional outcomes as percents

double[][] conditionalPercents = new double[6][6];

for (int i = 0; i < 6; i++) {

for (int j = 0; j < 6; j++) {

conditionalPercents[i][j]

= ((double) conditionalCounts[i][j] / counts[i]) \* 100;

}

}

// for each outcome 1-6, display the percent 1-6 came up on next roll

for (int i = 0; i < 6; i++) {

System.out.println("Given a roll of " + (i + 1) + ":");

for (int j = 0; j < 6; j++) {

System.out.printf(" The number %d came up next %.3f%% of "

+ "the time\n", (j + 1), conditionalPercents[i][j]);

}

}

}

/\*\*

\* This method rolls 2 dice a number of times the user chooses. It then displays

\* the frequency (as a percent) with which doubles were rolled.

\* @param keyboard for user input

\* @param dr the DiceRoller

\*/

public static void testCase3(Scanner keyboard, DiceRoller dr) {

System.out.println("\nTest Case 3");

System.out.println("This is a test for doubles when rolling 2 dice.");

System.out.print("How many times would you like to roll 2 dice: ");

int numRolls = keyboard.nextInt();

int[] rollTwo;

int doublesCount = 0;

// roll 2 dice the requested number of times, count # of times doubles rolled

for (int i = 0; i < numRolls; i++) {

rollTwo = dr.rollDice(2);

if (rollTwo[0] == rollTwo[1]) {

doublesCount++;

}

}

// calculate and display the percentage that doubles were rolled

double percentDoubles = ((double) doublesCount / numRolls) \* 100;

System.out.printf("The percent of times doubles were rolled was %.3f%%\n",

percentDoubles);

}

/\*\*

\* This method rolls 2 dice a number of times the user chooses. It then displays

\* the frequency (as a percent) with which each possible sum of 2 dice was rolled.

\* @param keyboard for user input

\* @param dr the DiceRoller

\*/

public static void testCase4(Scanner keyboard, DiceRoller dr) {

System.out.println("\nTest Case 4");

System.out.println("This is a distribution test for the sum of 2 dice.");

System.out.print("How many times would you like to roll 2 dice: ");

int numRolls = keyboard.nextInt();

int[] rollTwo;

int[] diceSums = new int[11];

int sum;

// roll 2 dice the requested # of times, count frequency of each sum

for (int i = 0; i < numRolls; i++) {

rollTwo = dr.rollDice(2);

sum = rollTwo[0] + rollTwo[1];

diceSums[sum - 2]++;

}

// calculate and display the percentage that each possible sum came up

double[] percentEachSum = new double[11];

for (int i = 0; i < 11; i++) {

percentEachSum[i] = ((double) diceSums[i] / numRolls) \* 100;

System.out.printf("The sum %d came up %.3f%% of the time\n",

(i + 2), percentEachSum[i]);

}

}

/\*\*

\* This method rolls a single die a number of times the user chooses, and

\* then it repeats the same number of rolls in a second trial. It then examines

\* the frequency with which each number came up in the two trials, and displays

\* the difference between those two frequencies.

\* @param keyboard for user input

\* @param dr the DiceRoller

\*/

public static void testCase5(Scanner keyboard, DiceRoller dr) {

System.out.println("\nTest Case 5");

System.out.println("This is a test to ensure two separate distributions "

+ "are not identical.");

System.out.print("How many times would you like to roll the die each time: ");

int numRolls = keyboard.nextInt();

double[][] percentRolled = new double[2][6];

// conduct the trial twice

for (int j = 0; j < 2; j++) {

int[] counts = new int[6];

int roll;

// roll die the requested number of times, track counts for each outcome

for (int i = 0; i < numRolls; i++) {

roll = dr.rollDice(1)[0];

counts[roll - 1]++;

}

// calculate the frequency (as a percent) of each outcome

for (int i = 0; i < 6; i++) {

percentRolled[j][i] = ((double) counts[i] / numRolls) \* 100;

}

}

// calculate and display the difference in the frequencies (as percents)

// between each outcome in trial 1 and the same outcome in trial 2

double difference;

for (int i = 0; i < 6; i++) {

difference = Math.abs(percentRolled[0][i] - percentRolled[1][i]);

System.out.printf("For a roll of %d, the difference in the frequency "

+ "between trial 1 and trial 2 is %.3f%%\n", (i + 1), difference);

}

}

}